REMARKS

Reconsideration of this application is respectfully requested. The following remarks are responsive to the Office Action mailed September 9, 2002.

Claims 1-18 are pending.

Claims 1-8, 17, and 18 stand rejected under 35 U.S.C. §103(a) as allegedly being unpatentable over Dzikewich et al., U.S. Patent No. 5,706,500 ("Dzikewich") in view of Horiguchi et al., U.S. Patent No. 6,073,157 ("Horiguchi").

Claims 9-16 stand rejected under 35 U.S.C. §103(a) as allegedly being unpatentable over Dzikewich in view of Matsuda et al., U.S. Patent No. 5,790,419 ("Matsuda").

To establish a **prima facie** case of **obviousness**, three basic criteria must be met. First, there must be some suggestion or motivation, either in the references themselves or in the knowledge generally available to one of ordinary skill in the art, to modify the reference or to combine reference teachings. Second, there must be a reasonable expectation of success. Finally, the prior art reference (or references when combined) must teach or suggest all the claim limitations. The teaching or suggestion to make the claimed combination and the reasonable expectation of success must both be found in the prior art, and not based on applicant's disclosure. In re Vaeck, 947 F.2d 488, 20 USPQ2d 1438 (Fed. Cir. 1991).

Claims 1-8, 17, and 18 are not rendered obvious under 35 U.S.C. §103(a) in view of Dzikewich and Horiguchi.

The Office Action has rejected claims 1-8, 17, and 18 under 35 U.S.C. §103(a) as allegedly being unpatentable over Dzikewich in view Horiguchi. For the present claims to be rendered obvious by Dzikewich in view of Horiguchi, the cited art must individually or in combination teach each and every feature of the present claims. Furthermore, there must be some motivation or suggestion to combine the cited art.

Claim 1 includes the following limitations:

> automatically detecting exit of a child application object;

automatically terminating a grandchild application object launched by the child application object;

attempting restart of the child application object; and

signaling an outcome of the restart to a parent application object that launched the child application object.
(Claim 1, emphasis added)

The teachings of Dzikewich in combination with Horiguchi fails to render the present claims obvious because neither individually or in combination teach or suggest automatically terminating a grandchild application object launched by the child application object, attempting a restart of the child application object, or signal an outcome of the restart to a parent application object that launched the child application object. Dzikewich teaches the operation of a business event processor for processing at least one unit of work. Each unit of work includes an application context and a script processing state. The script processing state indicates which unit of work has already been processed by the business event processor. The inclusion of the application context and the script processing state allows the business processor to restart where it left off after a process failure based on the last recently processed unit of work. In other words, after a system failure, the business event processor is restarted and the step level recovery script (restart script) is automatically run so that the units of work may be processed at a point from which the error occurred. Although Dzikewich, through the creation of a step level recovery script, automatically detects the exit of a child application object, such as the units of work, Dzikewich does not discuss the attempt to restart the child application object nor does Dzikewich discuss signaling an outcome of the restart to a parent application object that launched the child application object. In contrast, the only way Dzikewich can restart the child application (units of work) is by restarting the parent application (business processing unit). Further, since the parent application is restarted and the restart or recovery script is automatically executed to continue where the business event processor left off, there is no discrete signaling of an outcome of the attempt to restart the child application object. The entire system merely restarts.

Adding what is taught in Horiguchi fails to cure the deficiency of Dzikewich. Although Horiguchi does teach automatically terminating a grandchild application object (threads) launched by the child application object (enclave), it does not teach <u>automatically detecting the exit of the child application object, attempting to restart the child application object after the exit, and signaling an outcome of the restart to a parent application object that launched the child application object,</u> as recited in claim 1 of the present application. Horiguchi discusses the execution of processes where each may comprise of one or more enclaves. An enclave is defined as a logical run-time structure that supports the execution of a group of procedures and can have multiple threads. A thread is an execution construct consisting of synchronous invocations and terminations of invocation units (including procedures). In other words, the hierarchy of levels is as follows: at the top is processes, which in turn are made up of enclaves, which are in turn made up of threads which can be a group of procedures. It is important to note, however, that there is no hierarchal relationship among processes (column 3, ll. 34-35) and no hierarchal relationship among threads (column 7, ll. 7-9). Because Horiguchi only discusses processes, enclaves, and threads, and does not discuss the limitations as recited in claim 1 of the present invention, claim 1 is patentable over the cited art. For at least these reasons, the dependent claims 2-8 are also patentable over the cited art. Because independent claims 17 and 18 have substantially similar limitations as claim 1, the same arguments that applied to claim 1 also apply to claims 17 and 18. Therefore, for at least the reasons stated above, independent claims 1, 17 and 18 and all dependent claim limitations therefrom are patentable over the cited art.

<u>**Claim 9 is not rendered obvious under 35 U.S.C. § 103 (a) in view of Dzikewich and Matsuda.**</u>

Claim 9 includes the following limitations:

> a watchdog automatically to <u>detect exit of a child application</u> object; and
>
> an executor <u>automatically to terminate a grandchild application</u> object launched by the child application object, to <u>attempt restart of the child application</u> object, and to <u>signal an outcome of the restart to a parent application</u> object that launched the child application object. (Claim 9, emphasis added)

The teachings of Matsuda fail to cure the deficiencies of Dzikewich with respect to the common elements of claim 9 and claim 1. Specifically, Matsuda does not discuss an executor <u>automatically terminating a grandchild application</u> launched by a child application, the <u>attempted restart of the child application</u> object, or the <u>signaling the outcome of the restart</u> to a parent application object. Further, Matsuda does not teach a watchdog <u>automatically to detect the exit of a child application</u>. Instead, Matsuda discusses a system for determining when a microcomputer is in an abnormal state. Specifically, Matsuda discloses a watchdog timer that monitors a pulse train frequency from a controller within the microcomputer circuit. When the pulse train falls outside a specified frequency range, the microcomputer is in an abnormal state and the watchdog timer outputs a signal to the controller that causes the microcomputer to reset. First, Matsuda does not discuss, as in the present claims, a <u>child application object</u>, but instead discusses a piece of hardware, a microcomputer, that is capable of executing a child application object. Second, Matsuda does not detect an <u>exit</u> of a child application as recited in the present claims. Conversely, in the unlikely event a microcomputer is deemed a child application object, Matsuda merely detects an error in the microcomputer's state based on a pulse train frequency measurement and not an exit of the child application object. The microcomputer itself does not exit, it only outputs an erroneous pulse train.

Because Matsuda does not cure the deficiencies of Dzikewich in regards to elements substantially similar to claim 1 and only discloses a watchdog timer that measures the frequency of a pulse train for detecting an abnormal state of a microcomputer and not to automatically detect an exit of a child application, claim 9 and dependent claims 10-16 are patentable over Dzikewich in view of Matsuda.

Authorization is hereby given to charge our Deposit Account No. 02-2666 for any charges that may be due. Furthermore, if an extension is required, then Applicants hereby request such an extension.

Respectfully submitted,

BLAKELY, SOKOLOFF, TAYLOR & ZAFMAN LLP

Dated: 6/30, 2003

Mark R. Vatuone

Mark R. Vatuone
Reg. No.: 53,719

12400 Wilshire Blvd.
Seventh Floor
Los Angeles, CA 90025
(408) 947-8200